Below is **exactly** what you asked for:

A **concrete**, **side-by-side BEFORE/AFTER comparison** of your tracked person
(the one originally at **index 6**)
with and **without** the condition

numberScansRow > currentNumberScansRow

This shows precisely how your logs and booleans behave differently.

This is the clearest possible way to understand why this condition is essential.

🧍 **THE PERSON WE TRACK**

Person originally at **index 6** in:

0 1 0 0 1 0 1 0 0 1 …

↑ tracked person

🔵 **BEFORE (your current code — FULL logic enabled)**

The tracked person will exhibit multi-phase movement:

PASS 1

They move:

6 → 7 → 8 (blocked)

Your logs show:

4START POS: …

New Person has commenced movement at position: 6

Same person continuing move at position: 7

PERSON FINISHED MOVING AT POSITION: 8

Then movement stops for an entire pass (PASS 2).

PASS 3 (the KEY PASS)

Later the blockage clears and the person can move again:

8 → 9 → 10 (blocked)

Because:

numberScansRow = 3

currentNumberScansRow = 1

The condition is triggered:

numberScansRow > currentNumberScansRow (true)

So your logs correctly show **a new movement burst**:

3START POS: ...

New Person has commenced movement at position: 8

Same person continuing move at position: 9

PERSON FINISHED MOVING AT POSITION: 10

Exactly what you wanted:
A second, distinct **movement phase**.

What your BEFORE-code "understands"

This person:

- moved in PASS 1
- paused
- then moved again in PASS 3
- and that is recognized as a **separate movement phase**

Your logs reflect actual behavior:

START → CONTINUE → FINISH

(wait)

START → CONTINUE → FINISH

🔴 **AFTER (if you remove the condition)**

Let's now see what happens when we remove:

numberScansRow > currentNumberScansRow

This single deletion changes the **entire interpretation** of the person's journey.

⚠️ **The person still moves physically the same:**

PASS 1:

6 → 7 → 8

PASS 2:

(no movement)

PASS 3:

8 → 9 → 10

BUT the logs change dramatically.

❌ **PASS 1 (unchanged)**

You still get:

4START POS

New Person has commenced movement at position: 6

Same person continuing at 7

PERSON FINISHED MOVING at 8

All good.

❌ **PASS 2 (now problematic)**

Still no movement.
But **the code no longer marks** that "we are between phases."

So:

- currentNumberScansRow stays stuck at 1
- No flags indicate "we are done with last movement burst"

This sets up the failure in PASS 3.

❌ **PASS 3 (the key failure)**

The person moves again:

8 → 9 → 10

BUT WITHOUT the condition numberScansRow > currentNumberScansRow:

- Your code does **NOT** treat this as a new movement phase
- It sees no reason to trigger a new "START POS"
- It incorrectly thinks the person is still in their original movement sequence

Let's walk through the logic:

At index 8, where movement restarts:

Variables are:

hasPersonPrevMove = false

hasPersonMove = false

Then the code sees:

people[8] == 1 && people[9] == 0 → true

(hasPersonMove becomes true)

Now your code checks:

if (!hasPersonPrevMove && hasPersonMove)

This would normally trigger **"New Person..."**,
but because numberScansRow > currentNumberScansRow is removed:

❌ **The code incorrectly thinks this is a *continuation* of the old movement phase.**

Two side-effects:

❌ **SIDE-EFFECT 1: Wrong log printed**

Instead of:

3START POS

New Person has commenced movement at position: 8

You get:

2START POS

Same person continuing move at position: 8

Your code completely misses the restart.

❌ SIDE-EFFECT 2: numPeopleMoving is wrong in certain arrays

In this specific array it stays correct (8 movers),
but in more complex patterns:

- false continuations are recorded
- true new movers never get counted
- movement phases collapse into one big “continuation”

This breaks your analytics.

❌ SIDE-EFFECT 3: “fin […]” block does not behave correctly

Because:

- hasPopulatedPreviousPerson never toggles correctly
- no “new phase” is recognized
- the system does not enter the improvise path properly

So "fin […]" prints either:

- too much,
- or not at all,
- or in the wrong place.

❌ SIDE-EFFECT 4: You lose the temporal structure of movement

What *should* be:

PASS 1:

START → CONTINUE → FINISH

PASS 2:

(wait)

PASS 3:

START → CONTINUE → FINISH

Becomes:

PASS 1:

START → CONTINUE → FINISH

PASS 3:

CONTINUE → FINISH

The second start event disappears.
The restart is erased.

🔥 **FINAL SIDE-BY-SIDE SUMMARY**

Behavior	**BEFORE (your code today)**	**AFTER (condition removed)**
Recognizes PASS 3 restart	✓ YES	❌ NO
"3START POS" appears in PASS 3	✓ YES	❌ NO
"New Person has commenced movement" appears in PASS 3	✓ YES	❌ NO
"Same person continuing" after PASS 3	✓ correct	❌ printed incorrectly
Tracks movement bursts	✓ YES	❌ collapsed into single burst
Tracks temporal behavior accurately	✓ YES	❌ LOSSES
Distinguishes movement vs waiting phases	✓ YES	❌ NO
Complex analytics (numPeopleMoving, fin blocks)	✓ correct	⚠ sometimes wrong

---

💡 **In plain English**

Your condition

numberScansRow > currentNumberScansRow

is the *only* thing telling your code:

“This person stopped moving earlier
and now is starting again in a NEW movement cycle.”

Without it:

- You still move everyone to the end
- But your logs lie
- Your booleans misclassify events
- You lose all restart recognition
- All movement phases collapse into ONE
- You can no longer track meaningful movement analytics
- Your output loses its **event-driven** nature